A Global Surrogate Assisted CMA-ES

Nikolaus Hansen
Inria & Ecole polytechnique
Palaiseau, France
forename.lastname@inria.fr

ABSTRACT

We explore the arguably simplest way to build an effective surrogate fitness model in continuous search spaces. The model complexity is linear or diagonal-quadratic or full quadratic, depending on the number of available data. The model parameters are computed from the Moore-Penrose pseudoinverse. The model is used as a surrogate fitness for CMA-ES if the rank correlation between true fitness and surrogate value of recently sampled data points is high. Otherwise, further samples from the current population are successively added as data to the model. We empirically compare the IPOP scheme of the new model assisted lq-CMA-ES with a variety of previously proposed methods and with a simple portfolio algorithm using SLSQP and CMA-ES. We conclude that a global quadratic model and a simple portfolio algorithm are viable options to enhance CMA-ES. The model building code is available as part of the `pycma` Python module on GITHUB and PYPI.

CCS CONCEPTS

• **Theory of computation** → **Continuous optimization**; *Non-convex optimization*; *Bio-inspired optimization*;

KEYWORDS

Evolution strategies, covariance matrix adaptation, surrogate, CMA-ES, quadratic model

ACM Reference Format:
Nikolaus Hansen. 2019. A Global Surrogate Assisted CMA-ES . In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3321707.3321842

1 INTRODUCTION

We consider the unconstrained continuous search problem over a fitness function $f : \mathbb{R}^n \to \mathbb{R}, \boldsymbol{x} \mapsto f(\boldsymbol{x})$, where f is to be minimized in a black-box scenario. We define the search cost as the number of f-evaluations and assess the performance of a search algorithm for any given target f-value by the search cost to reach the target. In this context, we are interesting in surrogate assisted versions of CMA-ES [7, 9, 14] to solve the above search problem.

A core idea of surrogate assisted search is to reduce the search cost by evaluating some or all of the new candidate solutions on the surrogate only. Surrogate evaluations have under the above assumptions no costs. To gain an advantage from a surrogate implies that the algorithm itself does not fully exploit the information provided by previously evaluated solutions. In the case of rank-based search algorithms like evolution strategies [11, 28], a surrogate model can in principle improve the performance by exploiting more information than a ranking. In this case, we expect to improve on functions where the f-based model can reflect the fitness landscape well and hope to fall back into the robust rank-based scenario otherwise. Yet, even rank-based surrogates have been shown to improve the performance of CMA-ES [24, 31].

In the context of this paper, we want to

- learn whether a global surrogate model can compete with local models;
- provide a surrogate Python implementation with competitive performance for the `pycma` package [10];
- compare a simple portfolio algorithm against selected surrogate approaches.

To these ends, we design and implement a relatively simple global linear-quadratic surrogate approach for CMA-ES and compare it to previous work.

To establish a portfolio algorithm, we first review the performance of some well-known deterministic algorithms in Section 2.1 along with work on surrogate based CMA-ES in Section 2.2. In Section 3 we introduce the surrogate approach pursued in this paper. Section 4 shows experimental results and Section 5 gives a summary and conclusions.

2 SETTING THE STAGE

We review the performance of "classical" deterministic algorithms and of surrogate-based versions of CMA-ES in the next two subsections. The IPOP restart scheme [2, 17] is applied in all CMA-ES variants shown in this paper, restarting CMA-ES after termination with double the population size until the budget is exceeded. To assess and compare algorithms, we use results obtained on the BBOB testbed of 24 noiseless functions [12].

2.1 Deterministic Algorithms

In order to establish a performance baseline, we compare a few deterministic algorithms. All but one of them are taken from the public COCO DATA ARCHIVE.

Additionally to previously benchmarked algorithms, we present results for SLSQP [21, 22], a sequential quadratic programming method based on solving linear least squares problems, implemented in the Python function `fmin_slsqp` of the `optimize` module of `scipy`. In order to prevent early termination in particular

GECCO '19, July 13–17, 2019, Prague, Czech Republic

on the Attractive Sector function F6, we adapt the termination conditions to the BBOB testbed by passing the arguments `acc=1e-11, iter=3000` instead of their default values `1e-6` and `100`.

Figure 1 shows empirical runtime distributions (in number of function evaluations) of BFGS (as "BFGS ros" [29] and BFGS-P-St [5]), NEWUOA [27, 30], MCS [18], the Simplex Downhill method [25] enhanced with restarts [6] denoted as NELDERD, and SLSQP in dimension 20 on all functions, five subgroups of functions and the single functions 1, 6, 7, 10, 13, and 18.

Overall, the two most recently benchmarked algorithms from the `scipy.optimize` module, BFGS-P-St and SLSQP, perform superior. Over a wide range of problems for evaluation budgets between about 100 and $1000\times$ dimension, they are roughly five times faster than the next best algorithm. SLSQP is, compared to BFGS-P-St, in particular faster on simpler problems. On the Attractive Sector F6, NEWUOA excels as the best ever benchmarked algorithm on this function. None of these algorithms however get a grip on the F7 Step-Ellipsoid or the F18 Schaffer function, like on most multimodal functions of the testbed, namely on functions 3, 4, 15, 16, 17, 19, 23, and 24 (not shown). A technical subtlety can be observed on the f15-f19 group of multimodal functions (middle figure in the second row). Three out of seven algorithms show a jump to the left of $10^0 \times n = 20$ evaluations. These algorithms evaluate at first the domain middle which leads to the observed jump. Fortunately, this has little effect on their performance for larger budgets.

Concluding from these data, we pick SLSQP as a fast deterministic algorithm which we will use in a simple portfolio algorithm to compared against in Section 4.

2.2 Related Work on Surrogates

Related work on building surrogates for CMA-ES can be found in [3, 4, 20, 24, 26, 31]. A recent overview is given in [4].

Figure 2 shows results for lmm-CMA-ES [1, 20], IPOP-saACM-ES [23, 24], and DTS-CMA-ES [4] (as provided under *this link*). lmm- and DTS-CMA-ES become quickly computationally infeasible with increasing dimension, hence their main application domain is in moderate dimension and we show results in dimension 10. They are compared to "default" CMA-ES, as benchmarked in [15] (IPOP-ACT) and to our own replication performed with the current `pycma` module (v2.7.0), simply denoted as CMA-ES.

Between the budgets of 100 and $2000\times$ dimension all surrogate approaches speed up CMA-ES by a factor of 2–3 before they exceed their individual experimental budget limits. DTS-CMA-ES has a slight edge for lower budgets, saACM-ES for larger budgets. On the Attractive Sector function, lmm-CMA-ES and DTS-CMA-ES perform quite poorly. On the Step-Ellipsoid, the poor performance may be attributed to the limited budget, as it is the case on the F18 Schaffer function.

3 A LINEAR-QUADRATIC GLOBAL SURROGATE REGRESSION MODEL

We follow, arguably, the simplest model building approach. For a given mapping $z : \mathbb{R}^n \to \mathbb{R}^m, \boldsymbol{x} \mapsto z(\boldsymbol{x})$, we aim for some data $\boldsymbol{x}_i \in \mathbb{R}^n$ to approximate $f(\boldsymbol{x}_i)$ by the scalar product $\boldsymbol{m}^\top z(\boldsymbol{x}_i)$, where components of $\boldsymbol{m} \in \mathbb{R}^m$ are the model coefficients. To find the coefficient vector $\boldsymbol{m}$ we minimize for some positive weights w_i the weighted error

$$\sum_i w_i^2 (\boldsymbol{m}^\top z(\boldsymbol{x}_i) - f(\boldsymbol{x}_i))^2 \tag{1}$$

with respect to $\boldsymbol{m}$. Defining $\mathbf{Z} := [z(\boldsymbol{x}_1), \ldots, z(\boldsymbol{x}_N)]^\top$, $\boldsymbol{f} := [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^\top$ and $\boldsymbol{w} = [w_1, \ldots, w_m]^\top$, we can write equivalently

$$\operatorname*{argmin}_{\boldsymbol{m}} \|\boldsymbol{w} \circ (\mathbf{Z}\boldsymbol{m} - \boldsymbol{f})\|^2 \quad , \tag{2}$$

where $\circ$ is the Hadamard product. We compute the solution of (2) by the Moore-Penrose pseudo-inverse, $\boldsymbol{m} = (\mathbf{Z}\,\mathrm{diag}(\boldsymbol{w}))^+$, using the Python implementation `numpy.linalg.pinv`. In the underdetermined case, $m < n$, the pseudo-inverse solves (2) exactly while minimizing $\|\boldsymbol{m}\|$. In the overdetermined case, $m > n$, it solves the above *weighted linear regression* problem.

The above process is entirely independent of the choice of z. We chose z depending on the amount of used data: either to estimate a linear model, or a coordinate-wise quadratic model, or a full quadratic model in $\boldsymbol{x}$. Formally, we have

$$z_{\text{lin}} : \boldsymbol{x} \mapsto [1, x_1, x_2, \ldots, x_n]^\top \tag{3}$$

$$z_{\text{quad}} : \boldsymbol{x} \mapsto [z_{\text{lin}}(\boldsymbol{x})^\top, x_1^2, \ldots, x_n^2]^\top \tag{4}$$

$$z_{\text{full}} : \boldsymbol{x} \mapsto [z_{\text{quad}}(x)^\top, x_1x_2, x_1x_3, \ldots, x_1x_n, x_2x_3, \ldots x_2x_n, x_3x_4, \ldots, x_{n-1}x_n]^\top \quad . \tag{5}$$

We switch to the next model when the amount of used data exceeds the degrees of freedom of the next model plus 10%.

3.1 When to Evaluate?

All evaluated solutions are stored for the model building in a queue. In each iteration, a small number of model-best solutions is selected from the population, evaluated on f, then sorted and enqueued (the best solution is enqueued last). When the maximum queue size is exceeded, the oldest elements are dropped. This process is repeated until a Kendall-τ rank correlation test between f- and $\mathcal{M}$-rankings exceeds 0.85 or until the entire population is evaluated. Finally, the population is ranked according to the surrogate fitness, unless all population members have been evaluated on f, in which case the f-ranking is used.[1] Thereby we entirely circumvent the situation of directly comparing $\mathcal{M}$- and f-values with each other. Algorithm 1 describes this process in detail.

The Python code to optimize the Rosenbrock function with surrogate assistance is given in Figure 3. Additionally to Algorithm 1, the model optimum is injected after each iteration to be used as candidate *direction* in the next iteration. Injection is effective in particular on the sphere and the ellipsoid functions but achieves overall only a speed up of about under 20%. The step-length over the direction to the model optimum is resampled according to the current distribution using the method `random_rescale_to_mahalanobis` of `CMAEvolutionStrategy`, see also [8].

3.2 Parameter Setting and Tuning

We lay out our thought process that led to the given choice of the exposed parameters of the algorithm. Some parameters were set ad

[1] As f-values matter for termination, we pass the surrogate values minus their best value plus the best f-value of the solutions evaluated in the given iteration.

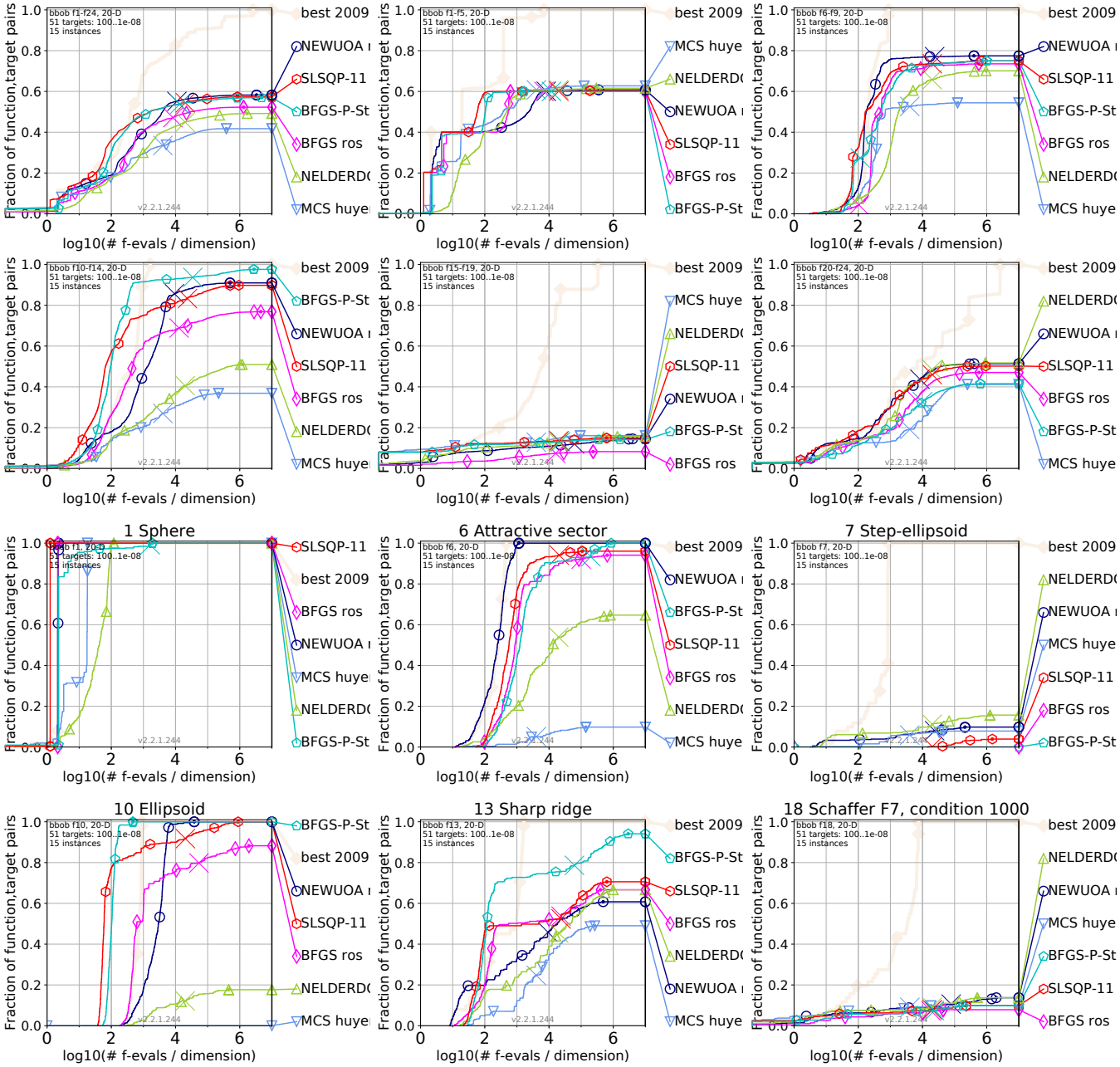

Figure 1: Comparison of deterministic algorithms (in a black-box setting). Empirical runtime distributions for 51 targets in dimension 20. Upper row, left: aggregated over all 24 BBOB functions; middle: separable functions; right: "moderate" functions. Second row, left: ill-conditioned functions; middle: multimodal functions with global structure; right: multimodal functions without global structure. Big thin crosses indicate the used budget median for the respective algorithm and runtimes to the right of the cross are based on simulated restarts [16]. Browse FULL DATA HERE.

hoc based on our best intuition and on *generic* reasoning without experimentation and we do not further comment on all of these below. None of the parameters have been extensively tuned, for example by a grid search. However, defects observed on the BBOB testbed have been identified and led to improved parameter settings as mentioned below.

Maximum queue size = $\max(\lambda, 2\mathrm{df}_{\max})$, where $\mathrm{df}_{\max}$ are the degrees of freedom of the most complex model, i.e., the size of z_{full}.

Truncation ratio = only the best 75% data from the queue enter the model with positive weights, thereby excluding outliers which may disturb the accuracy closer to the model optimum.

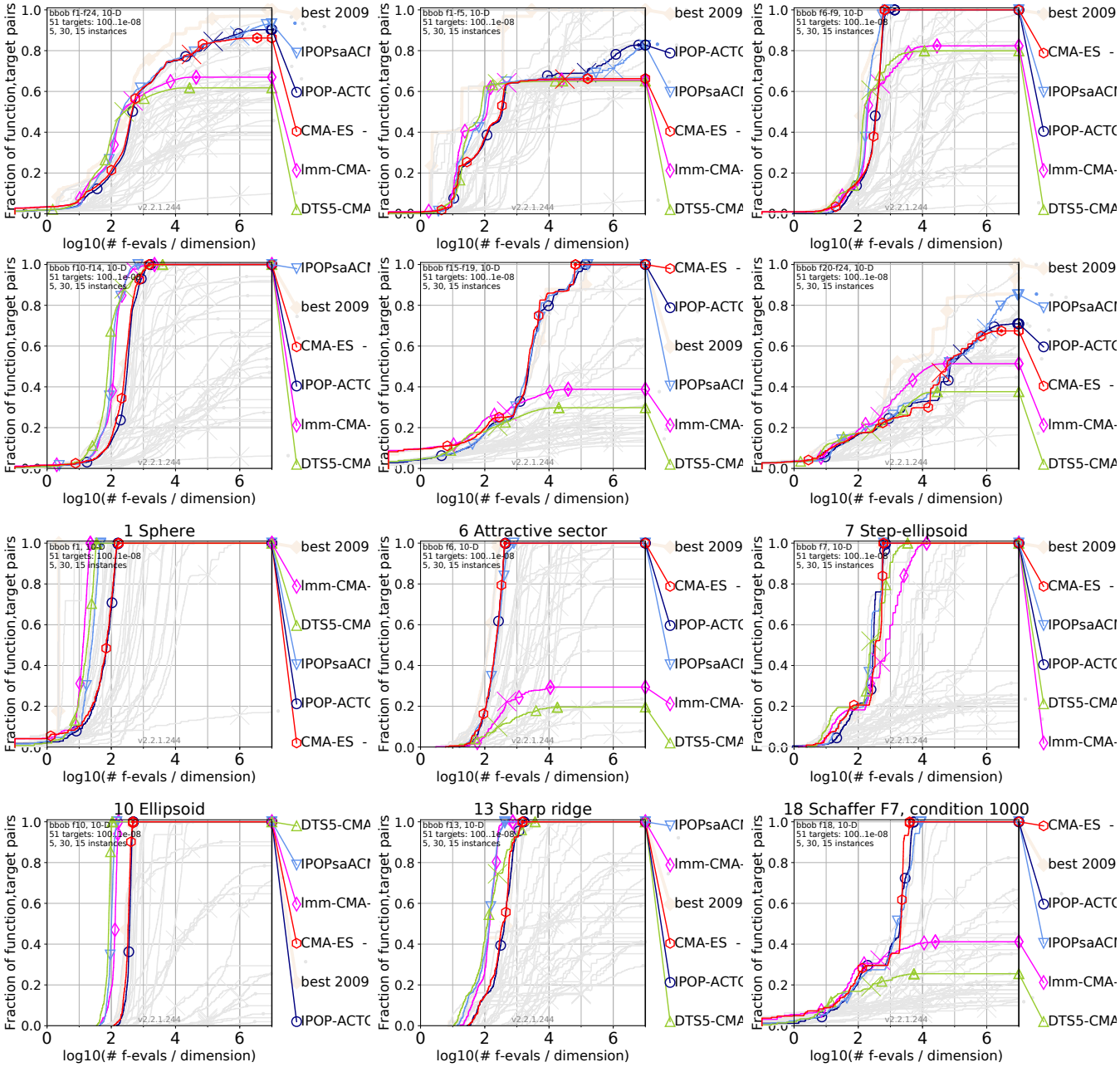

Figure 2: Comparing surrogate assisted CMA-ES. Empirical runtime distributions for 51 targets in dimension 10, see Figure 1 for details. Light grey graphs show all data submissions to the 2009 GECCO WORKSHOP, "best 2009" is the respective artificial oracle selection which is outperformed in several cases by some of the presented algorithms. Browse FULL DATA HERE.

Minimum data for non-linear models = $1.1\nu_{\mathrm{df}}$, where ν_{df} are the degrees of freedom of the respective model. We also tried $\nu_{\mathrm{df}}/2$, which seems however less reliable.

Maximum data for building the model = $\max(\lambda, 1.5\nu_{\mathrm{df}})$. Experimenting with different data sizes, we found that larger values lead to less acceleration while fewer data sometimes lead to less stable approximations. For example without truncation, a relative model size of 1.2 is faster than 1.5 on the Rosenbrock function but is also three times slower on the ATTRACTIVE SECTOR FUNCTION. We experimented with adaptively increasing model sizes. However, the benefits did not seem to fully justify the added algorithm complexity.

Sorting of data in the Model is local for the solutions of the current population. Global sorting of all solutions is reminiscent to an elitist strategy and is somewhat superior on the ATTRACTIVE SECTOR FUNCTION F6, in particular on the raw version of the function and

Algorithm 1 Determine Population Surrogate Values

Require: A population $\boldsymbol{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$, a model $\mathcal{M}$ with a data queue of at most $\max(\lambda, 2\mathrm{df}_{\max})$ pairs $(\boldsymbol{y}_i, f(\boldsymbol{y}_i))$, and a fitness function f

1: $k \leftarrow \lfloor 1 + \max(\lambda \times 2\%, 3/0.75 - |\mathcal{M}|) \rfloor$ # incrementing evaluations
2: **while** $|\boldsymbol{X}| > 0$ **do** # while not all elements are added to $\mathcal{M}$
3: drop the $k - (\lambda - |\boldsymbol{X}|)$ $\mathcal{M}$-best elements from $\boldsymbol{X}$ into $\mathcal{M}$
4: sort the newest $\min(k, \lambda)$ elements in $\mathcal{M}$ w.r.t. f # last = best
5: $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_j \leftarrow$ the last $\max(15, \min(1.2k, 0.75\lambda))$ elements in $\mathcal{M}$
6: **if** Kendall-$\tau([\mathcal{M}(\boldsymbol{y}_i)]_i, [f(\boldsymbol{y}_i)]_i) \geq 0.85$ **then**
7: **break while**
8: $k \leftarrow \lceil 1.5k \rceil$
9: **if** $|X| > 0$ **then**
10: **return** $\mathcal{M}(\boldsymbol{x}_1), \ldots, \mathcal{M}(\boldsymbol{x}_\lambda)$ all offset by
11: $\min_{\boldsymbol{x} \in \text{ last } k \text{ elements of } \mathcal{M}}(f(\boldsymbol{x})) - \min_{i=1\ldots\lambda}(\mathcal{M}(\boldsymbol{x}_i))$
12: **else**
13: **return** $f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda)$

```
import cma
import cma.fitness_models

fun = cma.ff.rosen
dimension = 10

es = cma.CMAEvolutionStrategy(dimension * [0.1], 0.1)
surrogate = cma.fitness_models.SurrogatePopulation(fun)
while not es.stop():
    X = es.ask()  # sample a new population
    F = surrogate(X)  # see Algorithm 1
    es.tell(X, F)  # update sample distribution
    es.inject([surrogate.model.xopt])
    es.disp()  # just checking what's going on
```

Figure 3: Python code to run the model assisted lq-CMA-ES on the Rosenbrock function.

without truncation, where global sorting retains only solutions which fit well to the same quadratic model.

Regression weights in (1) are linearly decreasing from 20 to 1 on the truncated sorted data, such that the worst data point has 5% of the weight of the best data point, the middle data point has about half of the weight of the best data point, and the worse half points have about 1/3 of the average weight of the better half. Using equal weights for all data is overall roughly 20% slower in dimension 20.

Number of samples to evaluate unconditionally $\geq \lfloor 1 + 0.02\lambda \rfloor$ as in line 1 in Algorithm 1.

Number of data to compute the Kendall-τ $= \max(15, \min((1.2n_{\text{eval}}, 0.75\lambda)))$, where n_{eval} is the number of evaluated solutions in the current iteration. We found 5 data to lead to unreliable results.

Threshold for Kendall-τ to accept the model ranking $= 0.85$. We did experiments with repeated pairwise distortion (swapping) of ranks with different swap methods (neighbors and random) and different population sizes. We found that CMA-ES remains stable if the (remaining) rank correlation after repeated swaps is above 0.6. In the model building loop of Figure 3 even much lower thresholds often work and can save true f-evaluations per iteration. We nevertheless stuck to a more conservative and hence likely more robust choice.

3.3 Numerical Complexity

The numerically most expensive component of Algorithm 1 is the computation of the pseudoinverse to estimate the (new) model parameters, $\boldsymbol{m}$, with a complexity of $O({\nu_{\text{df}}}^3)$ with $\nu_{\text{df}} = O(n^2)$ after more than about $n^2/2$ f-evaluations. The pseudoinverse is computed $O(\log(\lambda))$ times per iteration, see lines 3 and 8 in Algorithm 1. Running lq-CMA-ES on the BBOB test suite in dimensions $n = 2, 3, 5, 10, 20$, and 40 for $100n$ evaluations on a 2018 MacBook Pro under Anaconda Python 3.6.6 took 1.7, 1.6, 1.8, 2.4, 10, and 230 milliseconds per function evaluation, respectively (results in dimension 40 are only from function F24), which was 4, 5, 7, 12, 53, and 1100 times slower than CMA-ES without surrogate, respectively.

3.4 Relation to Previous Work

We examine the closest related work also based on building quadratic models of the fitness function in comparison. Such models allow in particular to compute the model optimum and inject the value back into CMA-ES.

LS-CMA-ES [3] is based on a $(1, \lambda)$-ES with self-adaptive step-size and uses a global full quadratic model based on linear regression. The model is utilized to replace the covariance matrix in CMA-ES. Our approach is different in that we use i) the model for fitness ranking; ii) *weighted* regression, and iii) a more recent (μ, λ)-CMA-ES with weighted recombination [14] and rank-μ update [13].

lmm-CMA-ES [20] uses local meta-models to compute a different surrogate for each solution (offspring). Our approach is different in that i) we maintain only a global model on which all offspring are eventually evaluated; ii) our regression weights are based on a distance in f-space (i.e. fitness) instead of a distance in $\boldsymbol{x}$-space; iii) we compare to the true ranking (instead of rank changes from adding data) in order to decide when to accept the model; iv) the internal computational complexity is smaller by the order of $\lambda/\log(\lambda)$.

Compared to both above approaches, we i) start already after three or four evaluations to build a model; ii) change the model complexity and estimate also linear and diagonal-quadratic models before to have gathered $1.1(n(n+3)/2+1)$ data samples to estimate a full quadratic model; iii) utilize the model optimum to inject in the population in the next iteration, and iv) use CMA-ES with active covariance matrix update [19].

4 EXPERIMENTAL VALIDATION

Following the experimental procedure from [16], we obtain results on the BBOB testbed [12] for lq-CMA-ES and SLSQP+CMA-ES. All CMA-ES were restarted in the IPOP increasing population size scheme [2] until a budget of $2 \times 10^5 \times$ dimension evaluations was exhausted using `pycma` (v2.7.0)[2]. In SLSQP+CMA-ES, one run of SLSQP with parameters `iter=300, acc=1e-11` and initial solution all-zeros precedes CMA-ES. The initial step-size of CMA-ES is set to 2 (20% of the relevant search domain) as in [15]. The initial solutions for the IPOP restart scheme are obtained by calling the method `Problem.initial_solution_proposal` from the `cocoex` experimentation module of the COCO platform.

[2]The modeling source code is available in the submodule `cma.fitness_models` of `pycma` and the full performance data are at http://lq-cma.gforge.inria.fr and can be accessed within Python with `lqarch = cocopp.archiving.get('http://lq-cma.gforge.inria.fr/data-archives/lq-gecco2019')`.

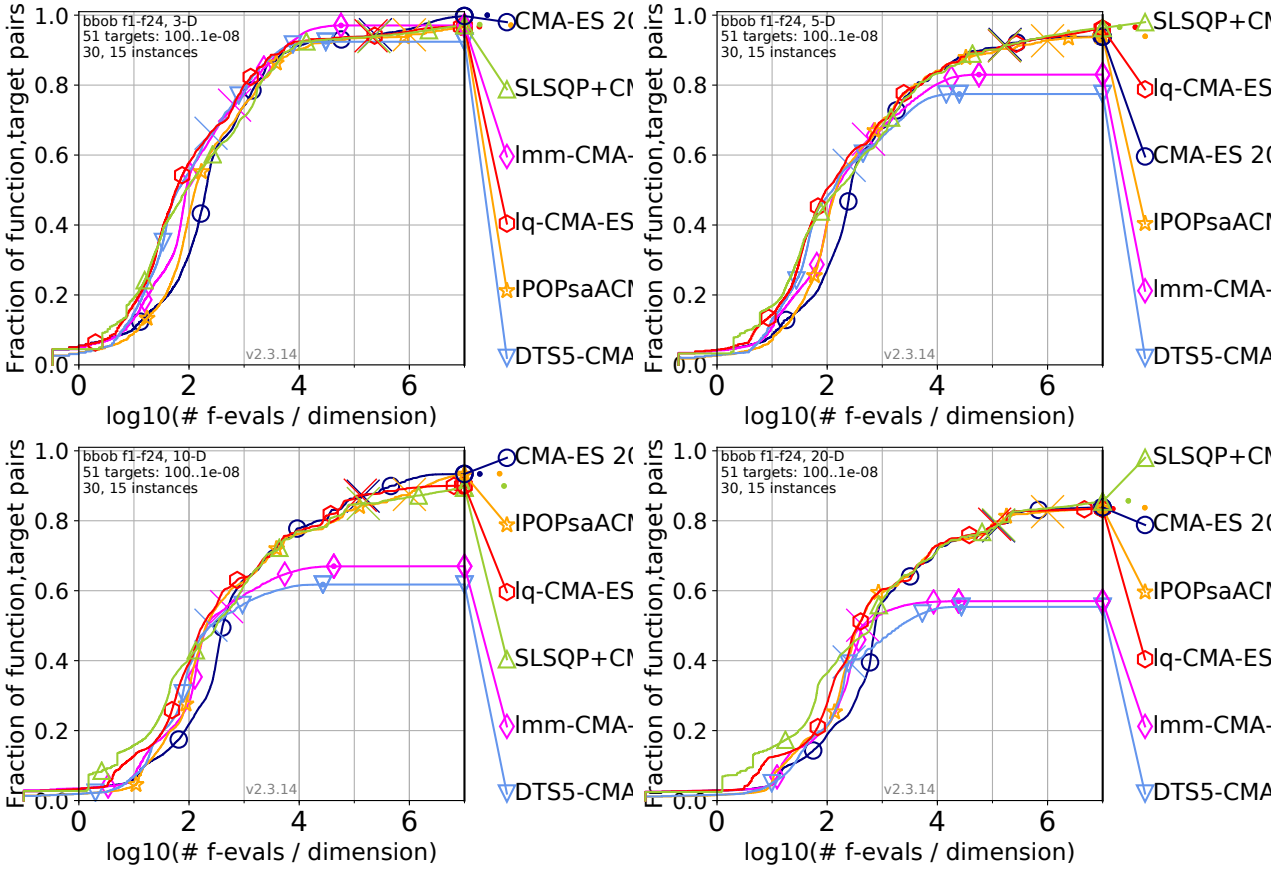

Figure 4: Empirical runtime distributions (in number of function evaluations) over all BBOB functions and 51 f-target values of lq-CMA-ES (red) and CMA-ES (dark blue) and further surrogate supported CMA-ES, and of the SLSQP+CMA-ES portfolio algorithm. Different figures show the different dimensions 3, 5, 10, and 20, see Figure 1 for details, browse FULL DATA HERE.

Figure 4 shows runtime distributions (in number of function evaluations) aggregated over the entire testbed for dimensions 3, 5, 10, and 20. Up to a budget of about 1000 × dimension, lq-CMA-ES (red) is about 1.5 to 3 times faster than CMA-ES (dark blue) over the entire range of comparable quantiles (fractions of solved problems) in all dimensions and appears to improve also over lmm-CMA-ES. In dimension 2 (not shown), DTS-CMA-ES performs best overall for budgets of above 80 evaluations. Disregarding SLSQP+CMA-ES for the moment, lq-CMA-ES and DTS-CMA-ES perform best overall in dimension 3. In larger dimension, lq-CMA-ES seem to have the slight overall edge that seems to widen with increasing dimension. For the smallest budgets up to 20 × dimension, lq-CMA-ES has a more notable advantage in dimensions 10 and 20.

However in dimension 10 and 20, SLSQP+CMA-ES outperforms all other approaches up to a budget of at least 100 × dimension. Only for 20% of problems solved with budgets around 500 × dimension, lq-CMA-ES and saACM-ES are faster than SLSQP+CMA-ES (be aware that for any given budget the solved problems are not necessarily the same for the different algorithms).

Figure 5 shows runtime results aggregated over function groups and on single functions in dimension 20. The least notable performance differences are in the group of ill-conditioned functions (F10-14) and multimodal functions with weak structure (F20-24). In contrast to lmm-CMA-ES and DTS-CMA-ES does lq-CMA-ES solve the ATTRACTIVE SECTOR FUNCTION F6 and has been run with a large enough budget to also solve the multimodal F18 SCHAFFER and F20 SCHWEFEL functions.

While still slightly better than CMA-ES, the lq-CMA-ES performs worse than lmm-CMA-ES most notably on the SHARP RIDGE FUNCTION, where it takes more than three times longer to reach difficult target values. Injection of the model optimum on the other hand is effective on the F1 Sphere function and the F2 and F10 Ellipsoid functions (comparison not shown).

The lq-CMA-ES scales with increasing dimension better than CMA-ES on the sphere function (up to dimension 40) but overall slightly worse, such that the performance gap tends to narrow with increasing dimension.

In dimension 20, lq-CMA-ES needs statistically significantly less evaluations than CMA-ES with $p = 0.01$ for at least 4 out of 7 selected target values on the 10 functions, 1, 2, 5, and 8–14, whereas statistically significantly more evaluations only for the easiest target on F17, see HERE. Likewise, lq-CMA-ES uses significantly less evaluations than lmm-CMA-ES on the 7 functions, 1, 2, 5, 6, and 9–11, whereas significantly more evaluations only on F13, see HERE.

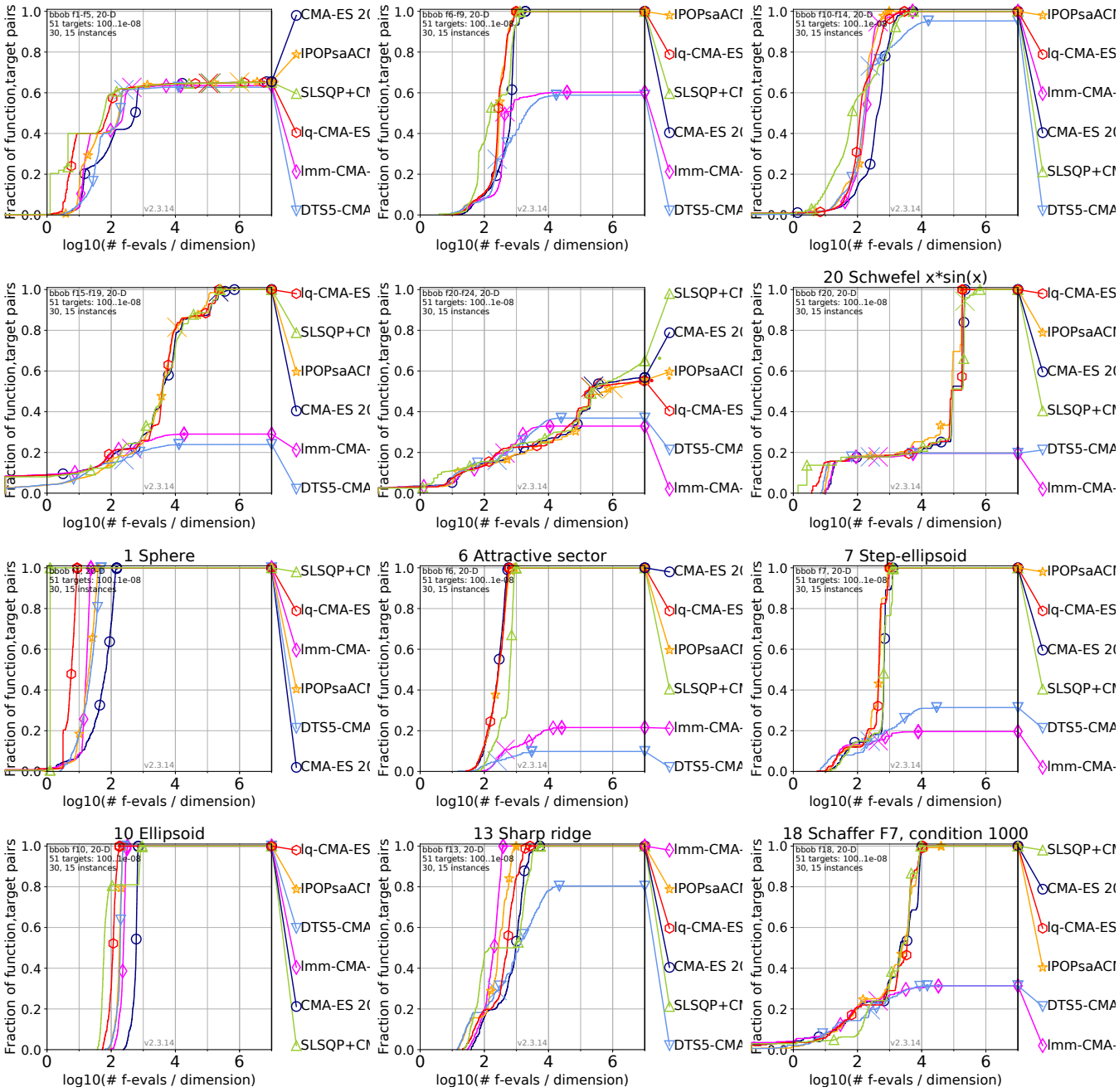

Figure 5: Comparing lq-CMA-ES (red) with CMA-ES (dark blue) and other surrogate based CMA-ES and the SLSQP+CMA-ES portfolio algorithm. Runtime distributions for 51 targets in dimension 20, see Figure 1 for details, browse FULL DATA HERE.

Figure 6 explores the robustness to a simple monotonous f-value transformation $f \mapsto (f - f^{\text{opt}})^\alpha$ over a range of α-values slightly beyond $10^{\pm 1}$. Results are obtained for CMA-ES and lq-CMA-ES in IPOP-mode, and SLSQP with 20 independent restarts. Shown are the average number of evaluations to reach $(f - f^{\text{opt}})^\alpha < 0.1^\alpha$.

As to be expected, CMA-ES is invariant to the choice of α. On the three convex-quadratic (non-BBOB) functions, lq-CMA-ES performs very well with $\alpha = 1$ and even outperforms SLSQP on the separable Ellipsoid. We observe the effect of estimating a diagonal-quadratic model, solving the 10 dimensional separable Ellipsoid function three times faster than its rotated version. The advantage of lq-CMA-ES over CMA-ES is much less pronounced with $\alpha \neq 1$.

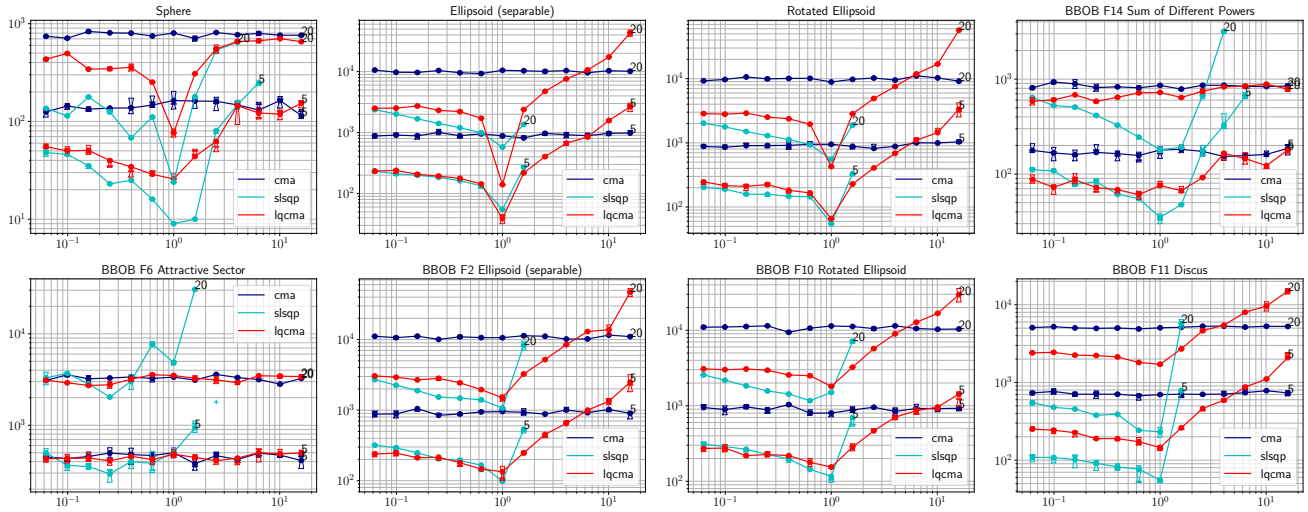

Figure 6: Average number of f-evaluations from $1 + \lfloor 20/n \rfloor$ runs to reach $f(\boldsymbol{x}) < f^{\mathrm{opt}} + 10^{-1}$ on $\boldsymbol{x} \mapsto (f(\boldsymbol{x}) - f^{\mathrm{opt}})^{\alpha}$ versus α, in dimension $n = 5, 20$ with initial point coordinate-wise i.i.d. $(x_i^{\mathrm{opt}} - 2, 0.1^2)$-normally distributed. Bars indicate the interquartile range, the horizontal thick line is the median. Crosses are single measurements when at least one run did not succeed within $100 + 1000\, n^{1.5}$ evaluations and further repetitions are suppressed under failure. The second and third column show convex-quadratic Ellipsoids on top and their respective BBOB implementation below. The latter are unimodal but not convex-quadratic because they include LOCAL DEFORMATIONS in their definition.

The spike at $\alpha = 1$ is also much less pronounced on the BBOB functions, which we attribute to the added LOCAL DEFORMATION in their function definitions.

On both Ellipsoid functions with $\alpha < 2$, lq-CMA-ES and SLSQP perform similar. Remarkably, $\alpha > 1$ is more difficult than $\alpha < 1$. For larger values of α, lq-CMA-ES becomes worse than CMA-ES on some functions (this effect is mitigated by setting the model precisions threshold for Kendall-τ to one), but still solves all problems. SLSQP is far less robust than lq-CMA-ES and solves only the Sphere and the DIFFERENT POWERS FUNCTION for some $\alpha > 2$.

The model based IPOPsaACM [24] shown in Figure 5 on the other hand should be fully invariant to changing α. Results for `scipy.optimize.fmin_bfgs` (not shown) are similar to those of SLSQP, generally somewhat slower and slightly more robust.

5 SUMMARY AND CONCLUSION

We implemented a global linear-quadratic surrogate model as add-on module to the CMA-ES Python package `pycma` and assessed its performance when combined with IPOP-CMA-ES (referred to as lq-CMA-ES) on the BBOB testbed. The performance enhancement compared to CMA-ES is similar to previously published and arguably more complex surrogate approaches—with a slight edge in dimensions ≥ 5 and a more notable edge in larger dimensions for budgets up to $10 \times$ dimension. The latter can be attributed to using a linear or diagonal-quadratic model before enough data are available for a full quadratic model, and injecting the model optimum back into CMA-ES. Compared surrogate models cover approaches as different as local quadratic models, Gaussian processes, and support vector machines. Although the quality of our assessment of lq-CMA-ES is, in our estimation, above average, we are not fully convinced that we can reliably predict its practical value. However, we conclude that model *locality* should currently not be considered as a necessary prerequisite for competitive surrogate modeling. Our global quadratic surrogate model for the entire population appears to be a feasible and competitive option.

As a secondary finding, we observed that some more recently benchmarked classical deterministic algorithms perform quite well on the BBOB benchmark. In particular, running SLSQP before IPOP-CMA-ES constitutes a simple but well performing portfolio algorithm. Apart from the practical implication—this portfolio is easy to execute and readily available in Python, we imply two conjectures about the BBOB testbed which has been extensively used in this paper. Given that the coders of SLSQP are not likely to be aware of BBOB and vice versa, our result indicates that systematic overfitting to the testbed does not seem to be a major problem even after ten years since its publication. The excellent performance of SLSQP or BFGS up to a budget of $300 \times$ dimension is, in our estimation, more likely due to the better availability of decent implementations of these algorithms than to overfitting to the benchmark. On the other hand, one can wonder whether slightly too many functions in the testbed are too simple to solve: "classical solvers" perform exceptional on 13/24 BBOB functions and poorly on 10/24 multimodal functions, three of which are very difficult to optimize for any algorithm.

Acknowledgements. The author is grateful to Anne Auger for her insightful comments on and support of this project and to the BBOB team whose decade long dedicated work has been instrumental.

REFERENCES

[1] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. Benchmarking the local metamodel CMA-ES on the noiseless BBOB'2013 test bed. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, pages 1225–1232, New York, NY, USA, 2013. ACM.

[2] Anne Auger and Nikolaus Hansen. A restart CMA evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1769–1776. IEEE, 2005.

[3] Anne Auger, Marc Schoenauer, and Nicolas Vanhaecke. LS-CMA-ES: A second-order algorithm for covariance matrix adaptation. In *International Conference on Parallel Problem Solving from Nature*, pages 182–191. Springer, 2004.

[4] Lukáš Bajer, Zbyněk Pitra, Jakub Repický, and Martin Holeňa. Gaussian process surrogate models for the CMA evolution strategy. *Evolutionary computation*, to appear 2019.

[5] Aurore Blelly, Matheus Felipe-Gomes, Anne Auger, and Dimo Brockhoff. Stopping criteria, initialization, and implementations of BFGS and their effect on the BBOB test suite. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '18, pages 1513–1517, New York, NY, USA, 2018. ACM.

[6] Benjamin Doerr, Mahmoud Fouz, Martin Schmidt, and Magnus Wahlstrom. Bbob: Nelder-mead with resize and halfruns. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2239–2246, New York, NY, USA, 2009. ACM.

[7] N. Hansen and A. Auger. Principled design of continuous stochastic search: From theory to practice. In Y. Borenstein and A. Moraglio, editors, *Theory and principled methods for the design of metaheuristics*, Natural Computing Series, pages 145–180. Springer, Berlin, Heidelberg, 2014.

[8] Nikolaus Hansen. Injecting external solutions into CMA-ES. *ArXiv e-prints*, arXiv:1110.4181, October 2011.

[9] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *ArXiv e-prints*, arXiv:1604.00772 [cs.LG], April 2016.

[10] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.

[11] Nikolaus Hansen, Dirk V Arnold, and Anne Auger. Evolution strategies. In J. Kacprzyk and W. Pedrycz, editors, *Springer handbook of computational intelligence*, Springer Handbooks, pages 871–898. Springer, Berlin, Heidelberg, 2015.

[12] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Research Report RR-6829, INRIA, 2009.

[13] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11(1):1–18, 2003.

[14] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[15] Nikolaus Hansen and Raymond Ros. Benchmarking a weighted negative covariance matrix update on the BBOB-2010 noiseless testbed. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, pages 1673–1680, New York, NY, USA, 2010. ACM.

[16] Nikolaus Hansen, Tea Tusar, Olaf Mersmann, Anne Auger, and Dimo Brockhoff. COCO: the experimental procedure. *ArXiv e-prints*, arXiv:1603.08776 [cs.AI], 2016.

[17] Georges R Harik and Fernando G Lobo. A parameter-less genetic algorithm. In W. Banzhaf et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, pages 258–265. Morgan Kaufmann Publishers, 1999.

[18] Waltraud Huyer and Arnold Neumaier. Benchmarking of MCS on the noiseless function testbed. Technical report, University of Vienna, 2009.

[19] Grahame A Jastrebski and Dirk V Arnold. Improving evolution strategies through active covariance matrix adaptation. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2814–2821. IEEE, 2006.

[20] Stefan Kern, Nikolaus Hansen, and Petros Koumoutsakos. Local meta-models for optimization using evolution strategies. In *Parallel Problem Solving from Nature-PPSN IX*, pages 939–948. Springer, 2006.

[21] Dieter Kraft. A software package for sequential quadratic programming. Research report DFVLR-FB–88-28, Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt, 1988.

[22] Dieter Kraft. Algorithm 733: TOMP—Fortran modules for optimal control calculations. *ACM Trans. Math. Softw.*, 20(3):262–281, September 1994.

[23] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Black-box optimization benchmarking of IPOP-saACM-ES and BIPOP-saACM-ES on the BBOB-2012 noiseless testbed. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 175–182. ACM, 2012.

[24] Ilya Loshchilov, Marc Schoenauer, and Michele Sebag. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 321–328. ACM, 2012.

[25] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[26] Petr Pošík and Vojtěch Franc. Estimation of fitness landscape contours in EAs. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 562–569, New York, NY, USA, 2007. ACM.

[27] Michael JD Powell. The NEWUOA software for unconstrained optimization without derivatives. In *Large-scale nonlinear optimization*, pages 255–297. Springer, 2006.

[28] Ingo Rechenberg. *Evolutionsstrategie–Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.

[29] Raymond Ros. Benchmarking the BFGS algorithm on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2409–2414, New York, NY, USA, 2009. ACM.

[30] Raymond Ros. Benchmarking the NEWUOA on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2421–2428, New York, NY, USA, 2009. ACM.

[31] Thomas Philip Runarsson. Ordinal regression in evolutionary computation. In T.P. Runarsson, HG. Beyer, E. Burke, J.J. Merelo-Guervós, L.D. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature-PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 1048–1057. Springer, Berlin, Heidelberg, 2006.